**Observations, Pros, and Cons of Each Method in Problem 1**

The swing equation was solved using several numerical methods, each with varying performance in terms of accuracy, stability, and computational efficiency. Below are observations and a comparison of the **ode45 (solve\_ivp)**, **Forward Euler**, **Backward Euler**, **Trapezoidal Rule**, **Euler’s Full-Step**, **Euler’s Half-Step**, and **Adam-Bashforth Second-Order** methods.

**1. ode45 (solve\_ivp with RK45)**

This is a Runge-Kutta method of order 4(5), which adapts its step size based on the required accuracy.

**Observations**:

* The **ode45** method is highly accurate and reliable for small and large time steps.
* The method takes variable step sizes to maintain accuracy, so it can automatically adjust to more challenging regions of the solution.
* With the step size set to 0.005s, the results are stable, and the error remains minimal.

**Pros**:

* **Adaptive step size** ensures accuracy without requiring manual time-step control.
* Handles stiff and non-stiff problems well.
* No significant manual intervention required.

**Cons**:

* **Computationally more expensive** than fixed-step methods, especially for larger simulations.
* May overestimate the number of time steps required, leading to longer computation times in some cases.

**2. Forward Euler's Method**

This is an explicit method that computes the next value based on the current value.

**Observations**:

* **Forward Euler** is simple to implement but has accuracy issues for larger step sizes.
* With the initial time step of 0.005s, the results are somewhat accurate, but as the step size increases (e.g., to 0.01s), the solution can become unstable.
* This method is highly sensitive to step size, and larger step sizes result in significant errors and divergence of the solution.

**Pros**:

* **Very simple to implement** and computationally cheap.
* Works well with small time steps.

**Cons**:

* **Stability issues** for larger time steps, leading to inaccurate solutions.
* Not suitable for stiff problems (requires very small time steps for stability).
* Convergence can be slow for small time steps, requiring many iterations.

**3. Backward Euler's Method**

This is an implicit method that solves for the next value using the current and next value.

**Observations**:

* **Backward Euler** is stable for larger time steps (e.g., 0.01s, 0.02s), even when Forward Euler fails.
* The method produces more accurate results than Forward Euler for larger step sizes, but it still suffers from accuracy loss when the time step becomes too large.
* The solution remains stable regardless of the time step size, but the method tends to be more **damped** and less responsive in capturing high-frequency changes in the solution.

**Pros**:

* **Unconditionally stable**, meaning it can handle larger time steps without blowing up.
* Works well for stiff problems.

**Cons**:

* Requires solving implicit equations, which can be computationally more expensive.
* **Lower accuracy** than explicit methods for the same step size, especially in non-stiff problems.

**4. Trapezoidal Rule Method**

This method takes the average of the Forward and Backward Euler methods, making it a second-order accurate method.

**Observations**:

* The **Trapezoidal Rule** strikes a good balance between accuracy and stability.
* For small time steps (0.005s), it provides a more accurate solution than both the Forward and Backward Euler methods.
* With larger time steps (e.g., 0.01s), it remains stable and accurate, outperforming both Forward and Backward Euler in terms of accuracy.

**Pros**:

* **Second-order accuracy**, leading to a more precise solution than either Euler method for the same step size.
* **Stable** for larger time steps.

**Cons**:

* Computationally more expensive than Forward Euler due to the need for implicit calculations.
* Less straightforward to implement than explicit methods.

**5. Euler’s Full-Step Modification Method**

This method uses a full-step prediction, which is a more direct extension of the Forward Euler method.

**Observations**:

* The **Full-Step Method** provides better accuracy than Forward Euler for small time steps (0.005s), but like Forward Euler, it becomes unstable for larger time steps.
* It performs better than Forward Euler in capturing the dynamics of the swing equation for small disturbances, but still suffers from accuracy issues when increasing the time step (e.g., 0.01s).

**Pros**:

* Slightly better accuracy than Forward Euler for small time steps.
* **Easy to implement** like the Forward Euler method.

**Cons**:

* **Instability** issues for larger time steps, just like Forward Euler.
* Not as accurate or stable as second-order methods (e.g., Trapezoidal Rule).

**6. Euler’s Half-Step Modification Method**

In this method, a half-step prediction is made and then used to correct the full-step value.

**Observations**:

* The **Half-Step Method** improves the accuracy of the Forward Euler method by making a prediction at the midpoint of the time step.
* It is more stable and accurate than both the Forward and Full-Step Euler methods for small time steps (0.005s).
* It remains more stable at slightly larger time steps (e.g., 0.01s) compared to the regular Euler methods.

**Pros**:

* **Improved accuracy** over standard Forward Euler for the same step size.
* Easy to implement with slight modifications to the Euler method.

**Cons**:

* Still subject to **stability issues** for larger time steps, although less so than the basic Euler method.
* Computationally more expensive than basic Euler, though cheaper than implicit methods.

**7. Adam-Bashforth Second-Order Method**

This method is a multi-step explicit method that uses two previous time steps to compute the next value.

**Observations**:

* The **Adam-Bashforth method** is more accurate than both Forward and Backward Euler for small time steps (0.005s) and performs comparably to the Trapezoidal Rule.
* The solution remains accurate and stable for slightly larger time steps (e.g., 0.01s), making it a good choice when higher accuracy is needed.
* However, because it is explicit, it can still become unstable for very large time steps (e.g., 0.02s or larger).

**Pros**:

* **Higher accuracy** than the Euler methods for the same time step, thanks to its second-order nature.
* Does not require solving implicit equations, making it more efficient than the Trapezoidal Rule.

**Cons**:

* Still **prone to instability** for large time steps, though it performs better than Forward Euler.
* Requires storing the two previous time steps, which increases memory usage slightly compared to single-step methods.

**General Observations on Time Step Variation:**

* **Small Time Steps (0.005s)**: All methods, except Forward Euler, are relatively accurate and stable. The error is small, and the solution captures the oscillatory behavior of the swing equation well. As expected, the higher-order methods (ode45, Trapezoidal Rule, Adam-Bashforth) give the best results.
* **Moderate Time Steps (0.01s)**: Forward Euler begins to show instability, while Backward Euler and Trapezoidal Rule remain stable. Euler's Full-Step and Half-Step methods start losing accuracy, but Adam-Bashforth remains reasonably accurate.
* **Larger Time Steps (0.02s)**: Forward Euler and even Euler's Full-Step method can become unstable, leading to divergent results. Backward Euler remains stable but shows damping effects, while Trapezoidal Rule and Adam-Bashforth still maintain reasonable accuracy, though some error begins to appear.

**Conclusion:**

* **ode45 (solve\_ivp)** is the most accurate and versatile, but it is computationally expensive.
* **Backward Euler** is very stable but tends to over-damp the solution.
* **Trapezoidal Rule** offers a good balance of accuracy and stability for larger time steps.
* **Euler’s Half-Step** is a nice improvement over Forward Euler but is still sensitive to step size.
* **Adam-Bashforth** is the best explicit method for both stability and accuracy, but like all explicit methods, it is sensitive to larger time steps.